Lab 3.3: Input Controls

Task 1: Text Fields

Every text field expects a certain type of text input, such as an email address, phone number, or just plain text. So it's important that you specify the input type for each text field in your app so the system displays the appropriate soft input method (such as an on-screen keyboard).

Beyond the type of buttons available with an input method, you should specify behaviors such as whether the input method provides spelling suggestions, capitalizes new sentences, and replaces the carriage return button with an action button such as a **Done** or **Next**. This lesson shows how to specify these characteristics.

1. Insert the following strings:

```
<string name="name">Name</string>
<string name="phone">Phone</string>
<string name="email">Email</string>
<string name="register">Register</string>
```

2. Create the following views.

3. Assign appropriate input type for each of the input data.

4. Test runs your app and all input controls.

Task 2: Radio Button

1. Create a new Project and insert the following string.xml:

```
<string name="app_name">MyRadioButton</string>
<string name="radio_male">Male</string>
<string name="radio_female">Female</string>
<string name="btn_display">Display</string>
```

2. Prepare the following layout:

```
<RadioGroup
    android:id="@+id/radioSex"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <RadioButton
        android:id="@+id/radioMale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/radio_male"
        android:checked="true" />

    <RadioButton
        android:id="@+id/radioFemale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/radio_female" />

</RadioGroup>

<Button
    android:id="@+id/btnDisplay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/btn_display"
    android:onClick="showMessage"/>
```

3. Modify the MyActivity class:

```
public class MyActivity extends Activity {

  private RadioGroup radioSexGroup;
  private RadioButton radioSexButton;
  private Button btnDisplay;

  @Override
  public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
  }

  public void showMessage(View v) {

         radioSexGroup = (RadioGroup) findViewById(R.id.radioSex);
         btnDisplay = (Button) findViewById(R.id.btnDisplay);

         int selectedId = radioSexGroup.getCheckedRadioButtonId();

                           // find the radiobutton by returned id
         radioSexButton = (RadioButton) findViewById(selectedId);

         Toast.makeText(this,
```

```
                    radioSexButton.getText(), Toast.LENGTH_SHORT).show();

  }

}
```

4. Run your app.

Task 3: Toggle Button

1. Create a new Project and prepare the following layout.xml:

```
    <ImageView
        android:id="@+id/imageViewIcon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

    <ToggleButton
        android:id="@+id/toggleButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:onClick="showPicture"
        android:text="@string/visible" />
```

2. Create following string.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        . . .
    <string name="app_name">ToggleButton</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="visible">Visible</string>
</resources>
```

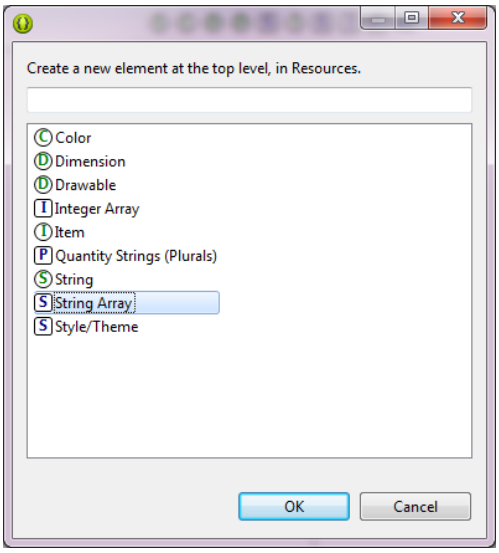
3. Modify the MyActivity class:

```
public class MyActivity extends Activity {
         ToggleButton toggleOnOff;
         ImageView imageViewIcon;

         @Override
         protected void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  setContentView(R.layout.activity_main);

                  imageViewIcon = (ImageView)findViewById(R.id.imageViewIcon);
         }

         public void showPicture(View v){
                  boolean on = ((ToggleButton)v).isChecked();

                  if(on)
                           imageViewIcon.setVisibility(View.VISIBLE);
                  else
                           imageViewIcon.setVisibility(View.INVISIBLE);
         }

}
```

4. Run your app.

Task 4: Spinner Button

1. Create a String Array in the res/values/string.xml:

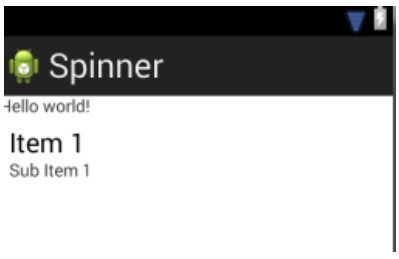


2. The string.xml should look like this:

```
<string name="app_name">Spinner</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
<string-array name="day">
    <item >Sunday</item>
    <item >Monday</item>
    <item >Tuesday</item>
    <item >Wednesday</item>
    <item >Thursday</item>
    <item >Friday</item>
    <item >Saturday</item>
</string-array>
```

3. Create the following layout:

```
<TextView
    android:id="@+id/textViewMessage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />

<Spinner
    android:id="@+id/spinnerDay"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```



4. Modify the MyActivity class:

```
public class MyActivity extends Activity implements AdapterView.OnItemSelectedListener{

	TextView textViewMessage;
	StringBuilder message;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		textViewMessage = (TextView)findViewById(R.id.textViewMessage);
		Spinner spinnerDay = (Spinner)findViewById(R.id.spinnerDay);

		spinnerDay.setOnItemSelectedListener(this);

		ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
				R.array.day, android.R.layout.simple_spinner_item);


	 adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

	spinnerDay.setAdapter(adapter);
	}


	@Override
	public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
	{
		textViewMessage.setText(parent.getItemAtPosition(position).toString());
	}

	@Override
	public void onNothingSelected(AdapterView<?> arg0) {
		// TODO Auto-generated method stub

	}

}
```

5. Run your app.

Task 5: Date Picker

---

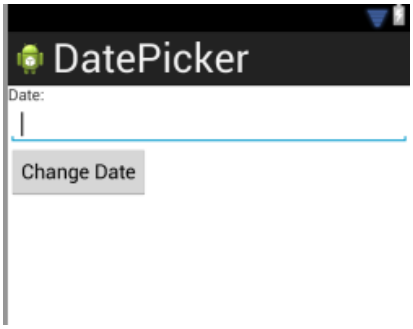
1. Create a new Project and prepare the following string.xml:

```
<string name="app_name">DatePicker</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
<string name="current_date">Date:</string>
<string name="change_date">Change Date</string>
```

2. Create the following layout:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/current_date" />

<EditText
    android:id="@+id/editTextDate"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="date" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/buttonChangeDate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/change_date"
        android:onClick="selectDate" />
```



3. Import the following support libraries:

   Because the DialogFragment class was originally added with Android 3.0 (API level 11), you need to include Support Library in your app. (Ignore this if your project is targeted at API 3.0 and above.)

```
import android.support.v4.app.DialogFragment;
import android.support.v4.app.FragmentActivity;
```

4. Modify the MyActivity class:

```
public class MyActivity extends FragmentActivity {

        EditText mEdit;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
        }

        public void selectDate(View view) {
                DialogFragment newFragment = new DatePickerFragment();
                newFragment.show(getSupportFragmentManager(), "DatePicker");
                //or
                // newFragment.show(getFragmentManager(), "DatePicker");
        }

        public void populateSetDate(int year, int month, int day) {
                mEdit = (EditText) findViewById(R.id.editTextDate);
                mEdit.setText(month + "/" + day + "/" + year);
        }
```

```
	@SuppressLint("ValidFragment")
	public class DatePickerFragment extends DialogFragment implements
			DatePickerDialog.OnDateSetListener {

		@Override
		public Dialog onCreateDialog(Bundle savedInstanceState) {
			final Calendar calendar = Calendar.getInstance();
			int yy = calendar.get(Calendar.YEAR);
			int mm = calendar.get(Calendar.MONTH);
			int dd = calendar.get(Calendar.DAY_OF_MONTH);
			return new DatePickerDialog(getActivity(), this, yy, mm, dd);
		}

		public void onDateSet(DatePicker view, int yy, int mm, int dd) {
			populateSetDate(yy, mm + 1, dd);

		}
	}

}
```

5. Run your app.

Exercises

Question 1

Write a mobile app to calculate life insurance premium using following data:

Age	Premium (RM)	Extra payment for male	Extra payment for smoker
Less than 5	50	0	0
6 – 10	55	0	0
11 – 16	60	50	0
17 – 21	70	100	100
22 – 30	120	100	150
31 – 40	160	50	150
41 – 50	200	0	250
More than 50	250	0	250

Use appropriate view and input type for all inputs.